Electronic Version
Stylesheet Version v1.1.1

Description

REDUNDANT EXTERNAL STORAGE VIRTUALIZATION COMPUTER SYSTEM

BACKGROUND OF INVENTION

[0001] 1. Field of the Invention

[0002] The invention relates to a redundant external storage virtualization computer system. More particularly, a redundant external storage virtualization computer system that uses point-to-point serial-signal interconnects as the primary device-side IO device interconnects is disclosed.

[0003] 2. Description of the Prior Art

[0004] Storage virtualization is a technology that has been used to virtualize physical storage by combining sections of physical storage devices (PSDs) into logical storage entities, herein referred to as logical media units (LMUs), that are made accessible to a host system. This technology has been used primarily in redundant arrays of independent disks (RAID) storage virtualization, which combines

smaller physical storage devices into larger, fault tolerant, higher performance logical media units via RAID technology.

[0005] An External (sometimes referred to as "Stand-alone") Storage Virtualization Controller is a Storage Virtualization Controller that connects to the host system via an IO interface and that is capable of supporting connection to devices that reside external to the host system and, otherwise, operates independently of the host.

[0006] One example of an external Storage Virtualization Controller is an external, or stand-alone, direct-access RAID controller. A RAID controller combines sections on one or multiple physical storage devices (PSDs), the combination of which is determined by the nature of a particular RAID level, to form logical media units that are contiguously addressable by a host system to which the logical media unit is made available. A single RAID controller will typically support multiple RAID levels so that different logical media units may consist of sections of PSDs combined in different ways by virtue of the different RAID levels that characterize the different units.

[0007] Another example of an external Storage Virtualization Controller is a JBOD emulation controller. A JBOD, short

for "Just a Bunch of Drives", is a set of physical DASDs that connect directly to a host system via one or more a multiple-device IO device interconnect channels. PSDs that implement point-to-point IO device interconnects to connect to the host system (e.g., Parallel ATA HDDs, Serial ATA HDDs, etc.) cannot be directly combined to form a "JBOD" system as defined above for they do not allow the connection of multiple devices directly to the IO device channel.

[0008] Another example of an external Storage Virtualization Controller is a controller for an external tape backup subsystem.

[0009] The primary motivation in configuring a pair of external storage virtualization controllers (SVCs) into a redundant pair is to allow continued, uninterrupted access to data by a host (or more than one host) even in the event of a malfunction or failure of a single SVC. This is accomplished by incorporating functionality into the SVCs that allow one controller to take over for the other in the event that the other becomes handicapped or completely incapacitated.

[0010] On the device side, this requires that both controllers are able to access all of the physical storage devices (PSDs) that are being managed by the SVCs, no matter which SVC

any given PSD may initially be assigned to be managed by. On the host side, this requires that each SVC have the ability to present and make available to the host all accessible resources, including those that were originally assigned to be managed by the alternate SVC, in the event that its mate does not initially come on line or goes off line at some point (e.g., due to a malfunction/failure, maintenance operation, etc.).

[0011] A typical device-side implementation of this would be one in which device-side IO device interconnects are of the multiple-initiator, multiple-device kind (such as Fibre, Parallel SCSI), and all device-side IO device interconnects are connected to both SVCs such that either SVC can access any PSD connected on a device-side IO device interconnect. When both SVCs are on-line and operational, each PSD would be managed by one or the other SVC, typically determined by user setting or configuration. As an example, all member PSDs of a logical media unit (LMU) that consists of a RAID combination of PSDs would be managed by the particular SVC to which the logical media unit itself is assigned.

[0012] A typical host-side implementation would consist of multiple-device IO device interconnects to which the host(s)

and both SVCs are connected and, for each interconnect, each SVC would present its own unique set of device IDs, to which LMUs are mapped. If a particular SVC does not come on line or goes off line, the on-line SVC presents both sets of device IDs on the host-side interconnect, its own set together with the set normally assigned to its mate, and maps LMUs to these IDs in the identical way they are mapped when both SVCs are on-line and fully operational. In this kind of implementation, no special functionality on the part of the host that switches over from one device/path to another is required to maintain access to all logical media units in the event that an SVC is not on-line. This kind of implementation is commonly referred to as "transparent" redundancy.

[0013] Redundant SVC configurations are typically divided into two categories. The first is "active-standby" in which one SVC is presenting, managing, and processing all IO requests for all logical media units in the storage virtualization subsystem (abbreviated SVS) while the other SVC simply stands by ready to take over in the event that the active SVC becomes handicapped or incapacitated. The second is "active-active" in which both SVCs are presenting, managing, and processing IO requests for the various

LMUs that are present in the SVS concurrently. In active-active configurations, both SVCs are always ready to take over for the other in the event that one malfunctions, causing it to become handicapped or incapacitated. Active-active configurations typically provide better levels of performance because the resources of both SVCs (e.g., CPU time, internal bus bandwidth, etc) can be brought to bear in servicing IO requests rather than the resources of only one SVC.

[0014] Another essential element of a redundant storage virtualization system is the ability for each SVC to monitor the status of the other. Typically, this would be accomplished by a implementing an inter-controller communications channel (abbreviated ICC) between the two SVCs over which they can exchange the operating status. This communications channel may be dedicated, the sole function of which is to exchange parameters and data relating to the operation of the redundant storage virtualization subsystem, or it can be one or more of the IO device interconnects, host-side or device-side, over which operational parameters and data exchange are multiplexed together with host-SVC or device-SVC IO-request-associated data on these interconnects.

[0015] Yet another important element of a redundant storage virtualization system is the ability of one SVC to completely incapacitate the other so that it can completely take over for the other SVC without interference. For example, for the surviving SVC to take on the identity of its mate, it may need to take on the device IDs that the SVC going off line originally presented on the host-side IO device interconnect, which, in turn, requires that the SVC going off line relinquish its control over those IDs.

[0016] This "incapacitation" is typically accomplished by the assertion of reset signal lines on the controller being taken off line bringing all externally connected signal lines to a pre-defined state that eliminates the possibility of interference with the surviving SVC. Interconnecting reset lines between the SVCs so that one can reset the other in this event is one common way of achieving this. Another way to accomplish this is to build in the ability of an SVC to detect when itself may be malfunctioning and "kill" itself by asserting its own reset signals (e.g., inclusion of a "watchdog" timer that will assert a reset signal should the program running on the SVC fail to poll it within a predefined interval), bringing all externally connected signal lines to a pre-defined state that eliminates the possibility

of interference with the surviving SVC.

[0017] Traditionally storage virtualization has been done with Parallel SCSI or Fibre IO device interconnects as the primary device-side IO device interconnects connecting physical storage devices to the storage virtualization controller pair. Both Parallel SCSI and Fibre are multiple-device IO device interconnects. Multiple-device IO device interconnects share bandwidth among all hosts and all devices interconnected by the interconnects.

[0018] Please refer to Fig.1, where a block diagram of a conventional redundant external storage virtualization computer system is illustrated. Note the interconnection of the host-side IO device interconnects that allows an SVC to take over for its mate by taking over the IO device interconnect IDs that would normally be presented onto the interconnect by its mate and mapping logical media units to these IDs in the same way its mate would. Also, note the interconnection of the device-side IO device interconnects that allow both SVCs access to all PSDs connected to the device-side IO device interconnects. In this example, a typical IO device interconnect that might be used on either host side or device side might be parallel SCSI or Fibre FC-AL, both multiple-initiator, multiple-device IO device

interconnects. Therefore, both SVCs operating in target mode (i.e., device mode) are connected to a single interconnect on the host side and allow both SVCs operating in initiator mode, together with multiple devices, to be interconnected on the device side. The configuration shown in Fig.1 suffers from the drawback that a malfunction of a single PSD, depending on the nature of the malfunction, can potentially bring down an entire device-side IO device interconnect making all other PSDs connected on the same interconnect inaccessible.

[0019] Fig.2 diagrams an improvement on this that effectively avoids the possibility that access to other PSDs connected on the same device-side IO device interconnect might be disrupted due to a malfunction that causes a single device-side interconnect to fail by making use of dual-ported PSDs and adding an additional interconnect to each PSD. In this way, the blockage of a single device-side IO device interconnect, possibly caused by a malfunction of an interconnect controller IC on the PSD, would not result in the inaccessibility of other PSDs connected on the same interconnect because the second interconnect connected to each of the same PSDs can be used to access those PSDs without interference.

[0020] The configuration shown in Fig.2 has the further advantage that IO request load can be distributed between the redundant device-side interconnects thereby effectively doubling the overall bandwidth of the device-side IO device interconnect subsystem as compared to the single-interconnect-per-PSD-set configuration shown in Fig.1. In this case, the typical device-side IO device interconnect of choice would typically be Fibre FC-AL because of the dual-ported nature of Fibre FC-AL PSDs currently on the market and the elements of the Fibre protocol that allow an initiator, such as an SVC, to determine which interconnect IDs on different interconnects correspond to the same PSD.

[0021] While the configuration depicted in Fig.2 is, indeed, far more robust than that depicted in Fig.1 in the face of device-side IO device interconnect failure, there is still the possibility that a PSD might malfunction in such a way that it could bring down both IO device interconnects that are connected to its dual-ported port pair. Were this to happen, once again, access to other PSDs connected on the same interconnect pair would be disrupted. In a logical media unit that consists of a standard singly-redundant RAID combination of PSDs (e.g., RAID 5), this

could prove disastrous for it can cause multiple PSDs in the combination to go off line causing the entire LMU to go off line.

SUMMARY OF INVENTION

[0022] It is therefore a primary objective of the claimed invention to provide a redundant external storage virtualization computer system using point-to-point serial-signal transmissions as the primary device-side IO device interconnects to solve the above-mentioned problem.

[0023] According to the claimed invention, a redundant external storage virtualization computer system is introduced. The redundant external storage virtualization computer system includes a host entity for issuing an IO request, a redundant storage virtualization controller pair coupled to the host entity for performing an IO operation in response to the IO request issued by the host entity, and a plurality of physical storage devices for providing storage to the computer system. Each of the physical storage devices is coupled to the redundant storage virtualization controller pair through a point-to-point serial-signal interconnect. The redundant storage virtualization controller pair includes a first and a second storage virtualization controller coupled to the host entity. In the redundant storage

virtualization controller pair, when the first storage virtualization controller is not on line or not in operation, the second storage virtualization controller will take over the functionality originally performed by the first storage virtualization controller. In one embodiment, of the present invention, the point-to-point serial-signal interconnect is a Serial ATA IO device interconnect.

[0024] It is an advantage of the claimed invention that in the redundant external storage virtualization computer system using Serial ATA as the primary device-side IO device, each physical storage device has a dedicated interconnect to the storage virtualization controller pair.

[0025] It is another advantage of the claimed invention that not only the payload data portion of information but also the control information are protected by the SATA IO device interconnect.

[0026] These and other objectives of the present invention will no doubt become obvious to those of ordinary skill in the art after reading the following detailed description of the preferred embodiment that is illustrated in the various figures and drawings.

Brief Description of Drawings

[0027] Fig.1 is a block diagram of a conventional redundant ex-

ternal storage virtualization computer system.

[0028] Fig.2 is a block diagram of another conventional redundant external storage virtualization computer system.

[0029] Fig.3 is a block diagram of a redundant external storage virtualization computer system according to the present invention.

[0030] Fig.4 is a block diagram of an access control switch using only one signal line to control states and a related truth table.

[0031] Fig.5 is a block diagram of an access control switch using two signal lines to control states and a related truth table.

[0032] Fig.6 is a block diagram of an SVC according to the present invention.

[0033] Fig.7 is an embodiment block diagram of the CPC in Fig.6.

[0034] Fig.8 is an embodiment block diagram of the CPU chipset/parity engine in Fig.7.

[0035] Fig.9 is a block diagram of the SATA IO device interconnect controller of Fig.6.

[0036] Fig.10 is a block diagram of the PCI-X to SATA controller of Fig.9.

[0037] Fig.11 is a block diagram of the SATA port of Fig.10

[0038] Fig.12 illustrates the transmission structure complying

with serial ATA protocol.

[0039] Fig.13 illustrates a first FIS data structure complying with serial ATA protocol.

[0040] Fig.14 illustrates a second FIS data structure complying with serial ATA protocol.

[0041] Fig.15 is a flow chart of a switchover process.

[0042] Fig.16 is a flow chart of a switchover process when a Binary Signal Pair Access Ownership Arbitration mechanism is used.

[0043] Fig.17 is a timing diagram for a switchover process of an alternate Binary Signal Pair Access Ownership Arbitration mechanism.

[0044] Fig.18 is a flow chart for the switchover process depicted in Fig.17.

[0045] Fig.19 is a flow chart of a mandatory switchover process when one SVC in the SVC pair malfunctions.

[0046] Fig.20 is a flow chart of an IO request routing.

[0047] Fig.21 is a block diagram of a redundant SVC interconnected expansion port implementation.

[0048] Fig.22 is a block diagram showing how hardware switches might be employed to achieve a switchable interconnection.

[0049] Fig.23 shows a circuit design that relies on hardware signal detection to activate the switch state changes.

[0050] Fig.24 shows a circuit design that takes input from the SVC1 and SVC2, *C1* and *C2* respectively, to trigger the switch state changes.

[0051] Fig.25 shows a hybrid circuit of those shown in Fig.23 and Fig.24.

[0052] Fig.26 is a block diagram of a redundant SVC interconnected redundant expansion port implementation.

[0053] Fig.27 is a block diagram of another redundant SVC interconnected expansion port implementation.

[0054] Fig.28 is a block diagram showing an implementation that uses hardware switches to interconnect the two IO device interconnects connecting the two SVCs to the storage units shown in Fig.27.

[0055] Fig.29 shows a circuit design that relies on hardware signal detection to activate the switch state changes of hardware switches shown in Fig.28.

[0056] Fig.30 shows a circuit design that takes input from the SVC1 and SVC2, *C1* and *C2* respectively, to trigger the switch state changes of hardware switches shown in Fig.28.

[0057] Fig.31 shows a hybrid circuit of those shown in Fig.29 and

Fig.30.

[0058] Fig.32 is a flow chart of an IO request routing in a redundant SVC interconnected expansion port implementation.

[0059] Fig.33 is a block diagram of a redundant external storage virtualization computer system comprising two separate host-side ports on each SVC connecting to two entirely separate host-side IO device interconnects and host ports.

[0060] Fig.34 is a block diagram showing an example of a switch circuit that can be used to accomplish thehost side interconnection of Fig.33.

[0061] Fig.35 is a block diagram of a redundant external storage virtualization computer system comprising one host-side port on each SVC connecting to one host-side IO device interconnect and host ports.

[0062] Fig.36 is a block diagram of a removable redundant SATA-PSD canister.

[0063] Fig.37 is a block diagram includes more details of the individual PCBs in the canister of Fig.36.

[0064] Fig.38 is a block diagram of a removable redundant PATA-PSD canister.

[0065] Fig.39 is a block diagram includes more details of the individual PCBs in the canister of Fig.38.

[0066] Fig.40 is a truth table for the control switch of Fig.4.

[0067] Fig.41 is a truth table for the control switch of Fig.5.

[0068] Fig.42 is a table for rerouting loop connections in the event of malfunction.

[0069] Fig.43 is a truth table for the circuit shown in Fig.29.

[0070] Fig.44 is a truth table for the circuit shown in Fig.30.

[0071] Fig.45 is a truth table for the circuit shown in Fig.31.

DETAILED DESCRIPTION

[0072] Please refer to Fig.3, where an embodiment block diagram of the current invention is illustrated.The system contains a host entity 10 and a redundant storage virtualization subsystem (SVS) 20. The SVS 20 contains a redundant storage virtualization controller pair (including SVC1 200 and SVC2 200) and a plurality of PSDs 420.The redundant storage virtualization subsystem configuration incorporates dedicated, point-to-point IO device interconnects to connect all PSDs 420 to both SVCs 200. The storage virtualization controller 200 can be a RAID controller or a JBOD emulator.

[0073] Although there is illustrated in Fig.3 only one host entity 10 connected with one SVS 20, there can be more than one host entity 10 attached to the SVS 20. The host entity

10 can be a host computer, such as a server system, a workstation, a PC system, or the like. Alternatively, the host entity 10 can be another SVC.

[0074] In order to allow both controllers to access the same PSD 420, an access control switch 342 is inserted in the device-side IO device interconnect path between the SVCs 200 and a PSD 420. Because of the point-to-point nature of the interconnect, only one SVC 200, i.e. the SVC 200 to which the particular PSD 420 is assigned at the time, can be actively accessing the PSD 420 at a time. The other SVC 200 remains in a stand-by mode with respect to this PSD 420 with its IO device interconnect to the particular PSD 420 disabled. One signal line from each SVC 200 is provided for controlling the access control switch 342. The switch 342 determines which SVC interconnect is patched through to the PSD 420.

[0075] As depicted in Fig.4, these signal lines may be wired together outside the switch to form a single control line that controls the state of the switch 342 according to a truth table shown in Fig.40. Alternately, as shown in Fig.5, the switch 342 can be designed to accept two control inputs, one from each SVC 200 with a truth table (Fig.41) determining the state of the switch 342 for the four possible

combinations of signals from the two SVCs.

[0076] In a redundant SVS in which it is important that any active components or groups thereof be positioned in hot-swappable units so that the subsystem does not have to be brought down in order to replace such components should a component failure occur. Such hot-swappable units are typically referred to as "Field Replaceable Units" (abbreviated FRU). Being active components, both the PSD itself and the access control switch would quite naturally also be located on FRUs. It makes sense to put them together on the same FRU, for one cannot achieve its intended functionality without the other. Therefore, the access control switch would typically be situated with the PSD in the removable PSD canister. Fig.36 and Fig.38 show block diagrams of one possible such arrangement.

[0077] In one implementation, all of the PSDs 420 in the SVS 20 can be combined to form a PSD array 400, and all the access control switches 342 can be combined to form a switching circuit 340. An example of such implementation is shown in Fig.6. Fig.6 depicts a block diagram showing an embodiment of an SVC 200 according to the present invention and the connection thereof to the host entity 10 and the PSD array 400. In this embodiment, the SVC1 200

comprises a host-side IO device interconnect controller 220, a central processing circuit (CPC) 240, a memory 280, a SATA IO device interconnect controller 300, and a redundant controller communicating (RCC) interconnect controller 236. Although illustrated in separate functional blocks, some or all of these functional blocks can be incorporated into to one chip. For example, the RCC interconnect controller 236 can be integrated with the host-side IO device interconnect controller 220 as a single-chip IC.

[0078] The host-side IO device interconnect controller 220 is connected to the host entity 10 and the CPC 240 to serve as an interface and buffer between the SVC1 200 and the host entity 10, and receives IO requests and related data from the host entity 10 and maps and/or transfers them to the CPC 240.

[0079] When the CPC 240 receives the IO requests of the host entity 10 from the host-side IO device interconnect controller 220, CPC 240 parses it and performs some operations in response to the IO requests and sends the data requested and/or reports and/or information of the SVC1 200 back to the host entity 10 through the host-side IO device interconnect controller 220.

[0080] After parsing a request received from the host entity 10, while a read request being received and performing one or more operations in response, the CPC 240 get the requested data either internally or from the memory 280, or in both ways, and transfers them to the host entity 10. If the data is not available either internally or does not exists in the memory 280, the IO request will be issued to the PSD array 400 through the SATA IO device interconnect controller 300 and the switching circuit. The requested data will then be transferred from the PSD array 400 to the memory 280 and passed to the host entity 10 through host-side IO device interconnect controller 220.

[0081] When a write request is received from the host entity 10, after parsing the request and performing one or more operations, the CPC 240 gets the data from the host entity 10 through the host-side IO device interconnect controller 220, stores them in the memory 280, and then transmits the data to the PSD array 400 through the CPC 240. When the write request is a write back request, the IO complete report can be issued to the host entity 10 first and then the CPC 240 performs the actual write operation later; otherwise, an IO complete report can be issued to the host entity 10 after the requested data is actually written into

the PSD array 400.

[0082] The memory 280 is connected to the CPC 240 and acts as a buffer to buffer the data transferred between the host entity 10 and the PSD array 400 through the CPC 240. In one embodiment, the memory 280 can be a DRAM; or more particularly, the DRAM can be a SDRAM.

[0083] The SATA IO device interconnect controller 300 is the device-side IO device interconnect controller connected between the CPC 240 and the PSD array 400. It serves as an interface and buffer between the SVC 200 and the PSD array 400 and receives IO requests and related data issued from CPC 240 and maps and/or transfers them to the PSD array 400. The SATA IO device interconnect controller 300 re-formats the data and control signals received from CPC 240 to comply with SATA protocol and transmits them to the PSD array 400.

[0084] An enclosure management service (EMS) circuitry 360 can be attached to the CPC 240 for management circuitry on an enclosure for containing the PSD array 400. In another arrangement of the SVS 20, the EMS circuitry 360 can be omitted, depending on the actual requirements of the various product functionality. Alternatively, the function of the EMS circuitry 360 can be incorporated into the CPC

240.

[0085] In this embodiment, the RCC interconnect controller 236 is implemented in SVC1 200 to connect the CPC 240 to SVC2 200. In addition, the SATA IO device interconnect controller 300 is connected to the PSD array 400 through the switching circuit 340. The switching circuit 340 is also connected to the SVC2 200. In this arrangement, the SVC2 200 can be attached to the SVC1 200. The PSD array 400 can be accessed by the two SVCs 200 through the switching circuit 340. Moreover, the control/data information from the host 10 can be transferred from the CPC 240 through the RCC interconnect controller 236 to the SVC2 200 and further to a second PSD array (not shown).

[0086] In Fig.7, an embodiment of the CPC 240 is shown, comprising the CPU chipset/parity engine 244, the CPU 242, a ROM (Read Only Memory) 246, a NVRAM (Non-volatile RAM) 248, an LCD module 350 and an enclosure management service circuitry EMS 360. The CPU can be, e. g., a Power PC CPU. The ROM 246 can be a FLASH memory for storing BIOS and/or other programs. The NVRAM is provided for saving some information regarding the IO operation execution status of the disk which can be examined after an abnormal power shut-off occurs and meanwhile

the IO operation execution does not complete. LCD module 350 shows the operation of the subsystem LCDs. EMS 360 can control the power of the DASA array and do some other management. The ROM 246, the NVRAM 248, the LCD module 350 and the enclosure management service circuitry EMS 360 are connected to the CPU chipset/parity engine 244 through an X-bus.

[0087] Fig.8 is a block diagram illustrating an embodiment of the CPU chipset/parity engine 244 according to the present invention. In the present embodiment, the CPU chipset/ parity engine 244 mainly comprises parity engine 260, CPU interface 910, memory interface 920, PCI interfaces 930, 932, X-BUS interface 940, and PM BUS 950. The PM BUS 950 is, for example, a 64-bit, 133Mhz bus and connects the parity engine 260, CPU interface 910, memory interface 920, PCI interfaces 930, 932, X-BUS interface 940 altogether for communicating data signal and control signal among them.

[0088] Data and control signals from host-side IO device interconnect controller 220 enter CPU chip/parity engine 244 through PCI interface 930 and are buffered in PM FIFO 934. The PCI interface 930 to the host-side IO device interconnect controller 220 can be, for example, of a band-

width of 64-bit, 66Mhz. When in the PCI slave cycle, the PCI interface 930 owns the PM bus 950 and the data and control signals in the PM FIFO 934 are then transmitted to either the memory interface 920 or to the CPU interface 910.

[0089] The data and control signals received by the CPU interface 910 from PM bus 950 are transmitted to CPU 242 for further treatment. The communication between the CPU interface 910 and the CPU 242 can be performed, for example, through a 64 bit data line and a 32 bit address line. The data and control signals can be transmitted to the memory interface 920 through a CM FIFO 922 of a bandwidth of 64 bit, 133 MHz.

[0090] An ECC (Error Correction Code) circuit 924 is also provided and connected between the CM FIFO 922 and the memory interface 920 to generate ECC code. The ECC code can be generated, for example, by XORing 8 bits of data for a bit of ECC code. The memory interface 920 then stores the data and ECC code to the memory 280, for example, an SDRAM. The data in the memory 280 is transmitted to PM bus 950 through the ECC correction circuit 926 and compared with the ECC code from the ECC circuit 924. The ECC correction circuit 926 has the functionality

of one-bit auto-correcting and multi-bit error detecting.

[0091] The parity engine 260 can perform parity functionality of a certain RAID level in response to the instruction of the CPU 242. Of course, the parity engine 260 can be shut off and perform no parity functionality at all in some situation, for example, in a RAID level 0 case. In one embodiment as shown in the Fig.8, the parity engine 260 can include an XOR engine 262 to connect with the PM bus 950 through XOR FIFO 264. The XOR engine 262 can perform, for example, the XOR function for a memory location with given address and length of the location.

[0092] The PLL (Phase Locked Loop) 980 is provided for maintaining desirable phase shifts between related signals. The timer controller 982 is provided as a timing base for various clocks and signals. The internal registers 984 are provided to register status of CPU chip/parity engine 244 and for controlling the traffic on the PM bus 950. In addition, a pair of UART functionality blocks 986 are provided so that CPU chip/parity engine 244 can communicate with outside through RS232 interface.

[0093] In an alternative embodiment, PCI-X interfaces can be used in place of the PCI interfaces 930, 932. Those skilled in the art will know such replacement can be easily ac-

complished without any difficulty.

[0094] Please refer to Fig.9, where an embodiment block diagram of the SATA IO device interconnect controller 300 of Fig.6 is illustrated. According to the present embodiment, the SATA IO device interconnect controller 300 comprises two PCI-X to SATA controllers 310. Fig.10 shows an embodiment block diagram of the PCI-X to SATA controller 310 of Fig.9. As shown in Fig.10, each PCI-X to SATA controller 310 comprises a PCI-X Interface 312 connected to the CPC 240, a Dec/Mux Arbiter 314 connected to the PCI-X interface 312, and 8 SATA Ports 600 connected to the Dec/Mux Arbiter 314. The PCI-X interface 312 comprises a bus interface 318 connecting to the Dec/Mux arbiter 314 and a configuration circuitry 316 storing the configuration of the PCI-X to SATA controller 310. The Dec/Mux arbiter 314 performs arbitration between the PCI-X interface 312 and the plurality of SATA ports 600 and address decoding of the transactions from the PCI-X interface 312 to the SATA ports 600. Through an SATA port 600 and the switching circuit 340, the data are transmitted to a PSD 420.

[0095] Next please refer to Fig.11. Fig.11 shows a block diagram illustrating an embodiment of the SATA port 600 of

Fig.10. As shown in Fig.11, the SATA ports 600 comprises a superset register 630, a command block register 640, a control block register 650, and a DMA register 620, all connected to the bus interface 318 of the PCI-X interface 312 through the Dec/Mux Arbiter 314. By filling these registers, data will be transferred between the Dec/Mux arbiter 314 and a transport layer 690 through a dual port FIFO 660 under the control of a DMA controller 670. The information received by a transport layer 690 will be re-formatted into a frame information structure (FIS) primitive and transmitted to a Link layer 700.

[0096] The Link layer 700 is then to re-format the FIS into a frame by adding SOF, CRC, EOF, etc., thereto and performing the 8b/10b encoding into encoded 8b/10b characters and transmits it to a PHY layer 710.

[0097] The PHY layer 710 will transmit signals through a pair of differential signal lines, transmission lines LTX+, LTX-, to and receive signals through another pair of differential signal lines, reception lines LRX+, LRX-, through the switching circuit 340 to a PSD controller in a PSD 420. The two signal lines of each pair of the signal lines, for example LTX+/LTX-, transmit signals TX+/TX- simultaneously at inverse voltage, for example, +V/-V or V/+V, with re-

spective to a reference voltage Vref so that the voltage difference will be +2v or 2V and thus enhance signal quality. This is also applicable to the transmission of the reception signals RX+/RX- on reception lines LRX+, LRX-.

[0098] When receiving a frame from the PHY layer 710, the Link layer 700 will decode the encoded 8b/10b characters and remove the SOF, CRC, EOF. A CRC will be calculated over the FIS to compare with the received CRC to ensure the correctness of the received information. When receiving a FIS from the Link layer 700, the transport layer 690 will determine the FIS type and distribute the FIS content to the locations indicated by the FIS type.

[0099] A transmission structure complying with serial ATA protocol is shown in Fig.12. The information communicated on the serial line is a sequence of 8b/10b encoded characters. The smallest unit thereof is a double-word (32 bits). The contents of each double-word are grouped to provide low-level control information or to transfer information between a host and an device connected thereto. Two types of data structures transmitted on signal lines are primitives and frames.

[0100] A primitive consists of a single double-word and is the simplest unit of information that may be communicated

between a host and a device. When the bytes in a primitive are encoded, the resulting pattern is not easy to be misinterpreted as another primitive or a random pattern. Primitives are used primarily to convey real-time state information, to control the transfer of information and to coordinate communication between the host and the device. The first byte of a primitive is a special character.

[0101] A frame consists of a plurality of double-words, and starts with an SOF (Start Of Frame) primitive and ends with an EOF (End Of Frame) primitive. The SOF is followed by a user payload called a FIS (Frame Information Structure). A CRC (Cyclic-Redundancy Check Code) is the last non-primitive double-word immediately proceeding the EOF primitive. The CRC is calculated over the contents of the FIS. Some other flow control primitives (HOLD or HOLDA) are allowed between the SOF and EOF to adjust data flow for the purpose of speed matching.

[0102] The transport layer constructs FISs for transmission and decomposes FISs received from the link layer. The transport layer does not maintain context of ATA commands or previous FIS content. As requested, the transport layer constructs an FIS by gathering FIS content and placing them in proper order. There are various types of FIS, two

of which are shown in Fig.13 and Fig.14.

[0103] As shown in Fig.13, a DMA setup FIS contains a HEADER in field 0. The first byte (byte 0) thereof defines the FIS type (41h), and the FIS type defines the rest fields of this FIS and defines the total length of this FIS as seven double-words. Bit D in byte 1 indicates the direction of the subsequent data transfer. D=1 means transmitter to receiver; D=0 means receiver to transmitter. Bit I in byte 1 is an interrupt bit. Bit R in byte 1 is a reserved bit and set to 0. DMA buffer identifier low/high field (field 1) indicates the DMA buffer region in the host memory. DMA buffer offset field (field 4) is the byte offset into the buffer. DMA transfer count field (field 5) is the number of bytes that will be read or written by the device.

[0104] As shown in Fig.14, a DATA FIS contains a HEADER in field 0. The first byte (byte 0) thereof defines the FIS type (46h), and the FIS type defines the rest fields of this FIS and defines the total length of this FIS as n+1 double-words. The R bits in byte 1 are reserved bits and set to 0. The fields 1 through n are double-words of data, which contain the data to transfer. The maximum amount of a single DATA FIS is limited.

[0105] Please refer back to Fig.4 and Fig.5. Typically, the access

control switch 342, of which the two depicted in Fig.4 and Fig.5 are examples, will be kept in a state that patches through to the PSD 420 the SVC 200 that is assigned to field device-side IO requests that are generated as a result of operations initiated in response to host-side IO requests. However, under certain conditions, it may be necessary to temporarily allow the alternate SVC 200 to access the PSD 420. An example of a configuration in which such a condition can arise is one in which one of the SVCs 200 is designated as the master when it comes to certain PSD-related management functions that are performed by the SVC 200 (e.g., monitoring the health of the PSD 420 or accessing an area of the media on the PSD 420 that is reserved for the SVC pairs internal use), while at the same time, some of the LMUs made accessible to the host entity 10 over the host-side IO device interconnects are assigned to the alternate SVC 200 for the purposes of host-side IO request processing. In this case, the SVCs 200 may communicate between themselves (possibly through the RCC interconnect controller 236 of Fig.6) to determine an appropriate opportunity in which the state of the switch 342 can be safely altered to allow the alternate SVC 200 access to the PSD 420 without disruption of PSD 420 ac-

cesses in process. This process will be referred to hereafter as "PSD access ownership transfer".

[0106] A switchover process is depicted in the flow chart in Fig.15. Each SVC 200 is able to determine the current state of the switch 342. This can be accomplished by allowing the SVC 200 to read back the state of the control signal(s) or by having each SVC 200 maintain an image in memory of the current state. The SVC 200 that is not currently patched through to the PSD 420 but requires PSD access, termed access requester, posts a request to the SVC 200 that is currently patched through to the PSD 420, termed access owner, to allow the state of the access control switch 342 to be switched to allow the access requester access to the PSD 420. On receipt of this request, the access owner waits for a "convenient" opportunity to start the switchover process. This process entails allowing any pending IO requests to complete while queuing any IO requests that have not yet started. When all pending requests have completed, the access owner modifies the state of its switch control signal to relinquish accessibility to the PSD 420 and then sends an acknowledgement to the access requester informing it that it can now safely access the PSD 420. At this point, the access requester

also modifies the state of its switch control signal to gain accessibility to the PSD 420 which completes the process of switchover and the access requester now becomes the new access owner.

[0107] At this point, the new access owner is free to issue IO requests to the PSD 420 at will. It may keep ownership until switching back of access ownership is requested by the original access owner, following the same procedure as above. Alternately, at some "convenient" opportunity, as when all the IO requests it has to issue for the time being are executed to completion, the new access owner can automatically start the switch-back process by modifying the state of its access control switch control signal to relinquish accessibility to the PSD 420 and issuing an unsolicited acknowledgement to the original access owner informing it that access ownership has been relinquished and the original access owner can now take back access ownership.

[0108] Typically, whether the new access owner keeps ownership until a request for switch-back from the original access owner is posted or it automatically transfers ownership back to the original access owner might be fixed by implementation or might be dynamically decided based on

such factors as relative frequency of access by the two SVCs 200 and the relative performance impact of keeping ownership versus automatically restoring ownership to the original access owner.

[0109] The mechanism of inter-controller communication to achieve the above switchover process can take any of a number of forms. One possible mechanism of communication, referred to here as the Binary Signal Pair Access Ownership Arbitration mechanism, is a pair of "access request" signal lines per SATA IO device interconnect, with each digital binary signal line having one SVC 200 set/clear (active SVC on this signal line) the signal and the other SVC 200 read the state of the signal (passive SVC). One of the access request signal lines has SVC1 200 active and SVC2 200 passive while the other access request signal line has SVC1 200 passive and SVC2 200 active. On the passive side, the SVC 200 can read both the current state of the alternate SVC"s access request signal and whether the signal has changed state since last reading. On reading, the latter will be cleared.

[0110] In this mechanism, at the outset, one SVC 200 has ownership of the SATA IO device interconnect and has the signal line on which it is actively asserted. When the alternate

SVC 200 wishes to assume ownership, it becomes the access requester and asserts the signal line on which it is active. It then monitors the signal line on which it is passive, watching for a change in its state indicating that it was deasserted at some point, which, in turn, indicates that the access owner acknowledged its request. At this point, the requesting SVC 200 can take control of the SATA IO device interconnect by altering the state of the switch 342 so that it is patched through to the PSD 420.

[0111] The access owner, on the other hand, continuously monitors for assertion the signal line on which it is passive. Following detection of assertion, at a "convenient" time, it starts waiting for any pending IO requests to complete while queuing any new IO requests. When all pending IO requests are complete, it acknowledges the access control request by deasserting the signal line on which it is active. If it wishes access control to be returned, as when there are new queued IO requests to be issued, it reasserts the signal line. The access requester monitors the signal line on which it is passive for a change in state rather than for a deasserted state because the access owner may have deasserted, then immediately asserted the signal line such that the access requester may not have the chance to de-

tect the deasserted state. Fig.16 depicts the flow chart described above.

[0112] A variation on the above *Binary Signal Pair Access Ownership Arbitration* mechanism for achieving coordinated access ownership transfer would be to implement a pair of HW circuits, referred to here as "Access Ownership Arbitration" circuits (abbreviated AOA), one for each SVC, that indirectly control the access control switch control signals rather than those signals being controlled directly by the SVC.

[0113] The output from one of the two *AOA*s would be connected to and control the access control switch control signal associated with one of the SVCs and the output from the other circuit would be connected to and control the access control switch control signal associated with other SVC. In addition, each of these *AOA*s would have the "Access Ownership Request" signals (abbreviated *AOR*) from both of the SVCs as inputs. When an SVC does not possess nor is requesting access ownership, its *AOR* is kept in a deasserted state. While in this state, the output signal of the *AOA* associated with this SVC is inactive. When the SVC wishes to assume ownership, it asserts its *AOR*. If the other SVC"s *AOR* is not active, then the *AOA* associated

with the requesting SVC would assert its output signal thereby asserting the access control switch control signal associated with the requesting SVC. If the other SVC"s *AOR* is active, then the requesting SVC"s *AOA*"s output remains deasserted until the other SVC"s *AOR* is deasserted, at which time the requesting SVC"s *AOA* output becomes active. The requesting SVC"s AOA output then remains active until the requesting SVC"s *AOR* is deasserted, independent of the state of the other SVC"s *AOR*. Typically, the two *AOA*s would be located in close proximity to the access control switch itself, such as in the PSD canister together with the access control switch.

[0114] A facility by which a SVC can determine whether or not it was granted access ownership in the event that both SVC"s assert their AORs concurrently and by which the SVC currently possessing access ownership can determine when the other SVC is requesting access ownership is also required in this mechanism. Providing a SVC the ability to determine the state of the access control switch would accomplish the former while providing the ability to determine the state of the other SVC"s *AOR* would achieve the latter. However, since these two determinations are made at different times during the access ownership transfer

process, they can be combined into a single facility consisting single digital binary signal per SVC, referred to here as the "Alternate SVC Access Ownership Request" signal (abbreviated *ASAOR*), the state of which can be read by the firmware running on the SVC. Normally, this signal would reflect the state of the other SVC"s *AOR*. However, when the SVC is granted access ownership, its *ASAOR* would cleared to inactive, independent of the state of the other SVC"s *AOR*, and remain in that state until read by the SVC firmware, after which it would go back to reflecting the state of the other SVC"s *AOR*. Fig.17 shows a timing diagram depicting the interaction between the various signals in this implementation.

[0115] In this mechanism, when a SVC wishes to assume ownership, it asserts it"s *AOR* and then starts monitoring its *ASAOR*. When the SVC detects that its *ASAOR* is inactive, it knows it has been granted ownership and can proceed to access the PSD. It then keeps its *AOR* asserted until it wishes to relinquish access ownership, at which point it deasserts its *AOR*. During the period in which the SVC wishes to maintain access ownership, it addition to keeping its AOR asserted, it also monitors its *ASAOR* for assertion. If it detects assertion indicating that the other SVC

wishes to assume ownership, at a "convenient" time, it would start waiting for any pending IO requests to complete while queuing any new IO requests. When all pending IO requests are complete, it would then relinquish ownership and deassert its *AOR*. If it wishes access ownership to be returned, as when there are new queued IO requests to be issued, it would immediately reassert its *AOR*. Fig.18 depicts the flow described above.

[0116] Another possible communication mechanism is passing access ownership transfer requests and acknowledgements over communication channels that support the transfer of multiple bits and/or bytes of information. A set of inexpensive dedicated communication channels, such as I2C channels, can be implemented for the purpose of exchanging these requests and acknowledgements. Alternately, the implementation can take advantage of the existing inter-controller communication channel (ICC) that allows the two SVCs 200 in the redundant pair to communicate with each other to exchange these access ownership transfer requests and acknowledges as part of the normal state-synchronization information that gets exchanged between the two SVCs 200.

[0117] A condition under which switchover of access ownership

would be mandated is when the access owner SVC 200 malfunctions in such a way that the alternate SVC 200 must take over its functions. Fig.19 diagrams the process of switchover of the access ownership in this case. On detection of malfunction of the malfunctioning SVC 200, the alternate SVC 200 asserts the malfunctioning SVCs reset signal to completely incapacitate it and to force all external signal lines into pre-defined states. One such external signal line is the access control switch control signal of the malfunctioning SVC 200. On assertion of the SVC reset signal, this signal line is set to a state that enables the patching through of the surviving SVC 200 to the PSD 420. Following the assertion of the malfunctioning SVC reset signal, the surviving SVC 200 sets the state of its access control switch control signal to engage patching through of itself to the PSD 420. This completes the switchover process.

[0118] The access control switch 342 will remain in this state until the malfunctioning SVC 200 is replaced or brought back on line and requests ownership to be transferred over to it. The state of the access control switch signal line for each controller at reset, power-up, and during initialization remains such as to disable patching through of

itself to the PSD 420 to insure that it does not interfere with potentially on-going PSD 420 accesses by the on-line SVC 200 by inadvertently forcing the access control switch 342 into a state that disrupts such accesses.

[0119] An alternate method of handling "occasional" access requirements on the part of the SVCs200 that does not normally have access ownership of the PSD 420 is to have the access owner act as an agent for issuing the IO requests that the SVC 200 requiring access, termed access requester, needs to have executed, an operation termed here as "IO Request Rerouting". This would typically entail transferring all the necessary IO request information to the access owner for it to construct into an IO request to issue to the PSD 420 for the access requester. In addition to the IO request information, the access requester would transfer any payload data to be written to the PSD to the access owner before or during IO request issuance and execution. Any payload data being read from the PSD would be transferred back to the access requester during or after IO request execution. Completion status of the operation, typically information that indicates whether the operation "succeeded" or "failed" and for what reason, would be passed back to the access requester on comple-

tion of the IO request execution. Fig.20 depicts such a flow chart.

[0120] There are a couple of advantages of IO Request Rerouting over actually transferring access ownership back and forth in order to allow both SVCs 200 to have access to each PSD 420. Firstly, because of the nature of Serial ATA protocol, requiring a rather extended "bring-up" on the interconnect in going from a "down" state to an "up" state, there could be a significant latency between when an access requester receives ownership from the access owner and when it can actually start initiating PSD accesses. Secondly, the process of "downing" the SATA interface then bringing it back up again may result in the need for SATA interface circuitry on either side to enter states of abnormal condition handling. Occasionally, because abnormal condition handling procedures are typically not as thoroughly tested as normal condition processing, bugs may appear which may interfere with the successful re-bring-up of the interconnect. To minimize this risk, it is good practice to try to minimize the occurrence of what either side would interpret and deal with as abnormal conditions, which, in this case would include minimizing the instances that access ownership needs be transferred.

[0121] One limitation of a "pure" Serial ATA SVC in which all of the device-side IO device interconnects are Serial ATA is that the number of PSDs that can be connected is limited by the number of device-side IO device interconnects that can be packed onto a single SVC. Because the SATA specification only allows for maximum signal line lengths of 1.5m, the PSDs connected to one SVC must be packed close enough so that no signal line length exceeds 1.5m. A typical SATA storage virtualization subsystem will only provide for connection of a maximum of 16 SATA PSDs because of these limitations. So a "pure" SATA storage virtualization subsystem is unable to match the expandability of a Fibre FC-AL storage virtualization subsystem, which would typically allow for connection of up to 250 PSDs via connection of external expansion chassis on the same set of device-side IO device interconnects.

[0122] In order to overcome this limitation, the current invention optionally includes one or more expansion device-side multiple-device IO device interconnects, herein referred to as device-side expansion ports, such as Parallel SCSI or Fibre FC-AL, on the SVC. These interconnects would typically be wired in such a way as to allow external connection of external expansion chassis. These chassis can be

simple "native" JBODs of PSDs directly connected to the interconnect without any intervening conversion circuitry or can be intelligent JBOD emulation subsystems that emulate "native" JBODs using a combination of SATA or PATA PSDs and a single or redundant set of SVCs that provide the conversion from the multiple-device IO device interconnect protocol that provides the connection of the JBOD subsystem to the primary storage virtualization subsystem to the device-side IO device interconnect (SATA or PATA) protocol that provides the connection between the JBOD SVC(s) and the PSDs that they manage.

[0123] The current invention introduces three possible options for wiring of the device-side expansion ports. Fig.21 depicts an implementation in which each device-side expansion port on one SVC is interconnected with its complement on the other SVC, referred to here as redundant SVC interconnected expansion port implementation. This allows both SVCs to share the device-side interconnect and the bandwidth it provides during normal operation and allows both SVCs full access to each storage unit port. It further allows either SVC to retain full access to all of the storage units, including those that were originally assigned to the alternate SVC, even in the event of an alter-

nate SVC malfunction.

[0124] Fig.22 shows how hardware switches might be employed to achieve this kind of switchable interconnection with an expansion port of a loop-style multiple-device IO device interconnect such as Fibre FC-AL. During normal operation, the state of all of the switches would be "0" thereby interconnecting both SVCs onto the interconnect with the storage unit. If SVC1 malfunctioned, M2 would be set to "1" while keeping M1 cleared at "0" thereby bypassing SV1 and creating a direct connection from SVC2 to the storage units. If SVC2 malfunctioned, M1 would be set to "1" (M2 is "Don"t Care") thereby bypassing SV2 and creating a direct connection from SVC1 to the storage units. This switching can be initiated by a hardware signal detection circuit (SDC) that detects whether or not there is a valid signal present on S1 or S2 or it can be initiated by one of the two SVCs when it detects a malfunction in the alternate SVC.

[0125] Fig.23 shows a circuit design that relies on hardware signal detection to activate the switch state changes. Fig.24 shows a circuit design that takes input from the SVC1 and SVC2, *C1* and *C2* respectively, to trigger the switch state changes. In this implementation, each control signal

would be forced into a CLEAR ("0") state when its corresponding SVC is off line (corresponding circuitry not shown in figure) to avoid the consequences that might arise should the control signal be tri-state or floating and thereby potentially be interpreted by follow on circuitry as SET ("1"). Fig.25 shows a hybrid of the two that supports switch state changes activated either by hardware signal detection or by input from the SVCs that offers greater flexibility than either one alone.

[0126] An enhanced such implementation is depicted in Fig.26, referred to here as the redundant SVC interconnected redundant expansion port implementation, would have pairs of redundant expansion ports rather than independent ports in order to keep a break or malfunctioning in an interconnect that services the expansion port of an SVC from causing a complete loss of access by the SVC to storage units connected on the interconnect. In such a configuration, each port in a redundant pair would connect to one of the ports in each of the dual-ported PSDs connected onto the interconnect or to one of the ports in a dual-ported storage virtualization subsystem that emulates a multiplicity of PSDs connected onto the interconnect (e.g., JBOD emulation storage virtualization subsys-

tem). Should one of the expansion ports or the connected interconnects malfunction on an SVC, IO requests would be rerouted through the other expansion-port/interconnect.

[0127] Fig.27 depicts another possible implementation in which each expansion port on one SVC has a redundant complement on the other SVC.The expansion port on one SVC and its redundant complement on the other SVC are connected to the two ports of each dual-ported storage unit in such a way that one SVC"s expansion port is connected to one port in the dual-ported pair and its complement on the other SVC to the other port. The complementary expansion ports are not interconnected but rather achieve redundancy by virtue of the dual-ported nature of each storage unit. However, dual-portedness of the storage units alone is not sufficient to support redundancy such that access by both SVCs can be maintained in the face of a malfunctioning expansion port on one of the SVCs or a malfunctioning interconnect connecting the expansion port on one of the SVCs to storage units. To achieve this, it is necessary to provide some mechanism for rerouting IO requests over to the interconnect that connects the alternate SVC to the storage units in the event that an SVC"s

own expansion port or the interconnect connecting it to the storage units malfunctions.

[0128] Fig.28 depicts an implementation that uses hardware switches to interconnect the two IO device interconnects connecting the two SVCs to the storage units shown in Fig.27 in the event of a break or malfunctioning of the portion of interconnect that runs between the SVS and the storage units. This portion of the interconnect would typically be a cable that runs between chassis and is especially vulnerable to breaks. In Fig.28, the state of all of the switches would be "0" during normal operation, that is, operation in which all interconnects are functioning properly. Signals from the expansion ports are routed directly to an associated port in the storage unit, referred to as the default path, so that each complementary expansion-port/interconnect in the redundant pair, one on each SVC, can operate completely independently without interference of any kind.

[0129] When it becomes necessary to route all IOs from both SVCs expansion ports to Storage Unit Port 1 due to a break in the interconnect connecting to Storage Unit Port 2 or perhaps a malfunction in Storage Unit Port 2 itself, M1 and M2 would be set to 1 while M3 would remain clear

at 0 and M4 and M5 are "Don"t Care". To route all IOs from both SVCs expansion ports to Storage Unit Port 2, M1 and M3 would be set to 1 while M4 and M5 would remain clear at 0 and M2 is "Don"t Care". If SVC1 goes off line, to route IOs from SVC2 directly to Storage Unit Port 2, M1, M4, and M5 would remain clear at 0 while M2 and M3 are "Don"t Care". If, in addition to SVC1 going off line there is also a break in the interconnect to Storage Unit Port 2 or perhaps a malfunction in the port itself, then IOs from SVC2 would need to be routed to Storage Unit Port 1. This would be done by setting M5 and M2 to 1 while M1 remains clear at 0 and M3 and M4 are "Don"t Care". Conversely, if SVC2 goes off line, to route IOs from SVC1 directly to Storage Unit Port 1, M2 and M3 would remain clear at 0 while M1, M4, and M5 are "Don"t Care". If, in addition to SVC2 going off line there is also a break in the interconnect to Storage Unit Port 1 or perhaps a malfunction in the port itself, then IOs from SVC1 would need to be routed to Storage Unit Port 2. This would be done by setting M3 and M4 to 1 while M5 remains clear at 0 and M1 and M2 are "Don"t Care". The table shown in Fig.42 summarizes the switch settings for the various possible scenarios.

[0130] The switching could be initiated by a hardware signal detection circuit (*SDC*) that detects whether or not there is a valid signal present on *S1* or *S2* or it could be initiated by one of the two SVCs when it detects a break or malfunction in the default path. Fig.29 shows a circuit design that relies on hardware signal detection to activate the switch state changes (a truth table for the circuit is shown in Fig.43). In this figure, *R1* is an override signal that allows SVC1 to forcibly take control of the state of the switches in the event that SVC2 and/or associated circuitry malfunction. Similarly, *R2* is an override signal that allows SVC2 to forcibly take control of the state of the switches in the event that SVC1 and/or associated circuitry malfunction. Fig.30 shows a circuit design that takes input from the SVC1 and SVC2, *C1* and *C2* respectively, to trigger the switch state changes(a truth table for the circuit is shown in Fig.44). In this case, setting C1 by SVC1 is an indication that its IO requests should be routed through Storage Unit Port 2 instead of the default Storage Unit Port 1 while setting C2 by SVC2 is an indication that its IO requests should be routed through Storage Unit Port 1 instead of the default Storage Unit Port 2. Fig.31 shows a hybrid of the two that supports switch state changes acti-

vated either by hardware signal detection or by input from the SVCs that offers greater flexibility than either one alone(a truth table for the circuit is shown in Fig.45).

[0131] Yet another option for wiring of device expansion ports in the configuration depicted in Fig.27 is without any interconnection at all. In this case, redundancy can be achieved by rerouting IO requests from one SVC to the other and out the surviving complementary expansion-port/device-side-interconnect over the inter-SVC communication interconnect that is normally used for synchronizing the mutual states of the two SVCs with each other.

[0132] When an SVC detects that a storage unit connected on an IO device interconnect that connects to one of its expansion ports can no longer be accessed, whether it is due to a detected break/malfunction in the expansion-port/interconnect or some other cause, the detecting SVC passes the IO request to the alternate SVC for the alternate SVC to issue to the same storage unit via the complementary expansion-port/interconnect and alternate storage unit port. Any data/status associated with the IO request is transferred between the two SVCs during the execution of the IO request. If the expansion-

port/interconnect on the alternate SVC appears to be up and functioning normally yet access to the storage unit fails on the alternate SVC also, the storage unit would be considered as having failed or having been removed. If access succeeds, then the loss of access would be considered to be localized to the original SVC and IO requests associated with future accesses to the storage unit are automatically rerouted to the alternate SVC for issuance over the complementary expansion-port/interconnect. During this time, the original SVC monitors the accessibility of the storage unit via its expansion-port/interconnect typically by periodically issuing internally generated IO requests that check the state of the interconnect and storage unit. If, at some point, the original SVC discovers that the storage unit can now be accessed over its expansion-port/interconnect, it will stop rerouting IO requests to the alternate SVC and start issuing them directly over its own expansion-port/interconnect again. Fig.32 shows a flow chart of this process.

[0133] Another feature that an SVC might typically implement is redundancy in the host-side interconnects in which multiple host-side interconnect ports are included on the SVC and LMUs are presented to the host identically over two or

more of these interconnects. This feature is designed to allow the host the ability to maintain access to the LMU even if one of the interconnects and/or ports on the interconnect should break, become blocked, or otherwise malfunction.

[0134] Fig.33 depicts a redundant external storage virtualization computer system comprising two separate host-side ports on each SVC connecting to two entirely separate host-side IO device interconnects and host ports. Each port on one SVC has a complementary port on the alternate SVC to which it is interconnected. In a typical implementation supporting redundancy in the host-side interconnects, each SVC would present the same set of logical media units in an identical fashion on both of its ports.

[0135] Under normal operation, host(s) can access logical media units through an SVC that is configured to present the LMU over a host-side interconnect. This can be one SVC or both of the SVCs in the redundant pair. If one SVC were to malfunction, logical media units that were already being presented to the host(s) by both SVCs would remain accessible through the normally-functioning SVC and, with the help of special purpose "multiple-redundant-pathing" functionality on the host, on detection that

IO request processing through one of the SVCs is disrupted, the IO requests would be completely routed to the normally-functioning SVC.

[0136] Those LMUs that were originally only being presented to the host by the SVC that is now malfunctioning would immediately be presented to the host(s) by the normally-functioning SVC over host-side interconnects that connect it to the hosts. For these LMUs, the normally-functioning SVC would be able to transparently take over the processing of host IO requests simply by presenting itself on each interconnect, together with all the reassigned logical media units, in an identical way to what the malfunctioning SVC did prior to its malfunctioning. With this kind of "transparent takeover", the host need not implement special functionality to make it aware of the SVC malfunctioning and reroute IOs itself in response.

[0137] In addition to SVC redundancy, the two sets of complementary ports in turn form a redundant port complement. A host that has two independent ports connected using two separate IO device interconnects to these two complementary redundant port sets then has two independent paths to each logical media unit over which it can issue IO requests. Should one of the ports on the host or on an

SVC malfunction or should the IO device interconnect itself break or become blocked, the hosts implementing multiple-redundant-pathing functionality can reroute IO requests over the other redundant path. Alternately, when both paths are functioning normally, the host can elect to issue IO requests over both paths in an effort to balance the load between the paths, a technique referred to as "load balancing".

[0138] To achieve the transparent takeover functionality described above, each of the pair of ports, one on each SVC, that form a complementary port pair are physically interconnected. For bus-style multiple-device IO device interconnects such as Parallel SCSI, the interconnection simply consists of wiring the devices together directly without any intervening circuitry. For other types of interconnects, special switch circuitry may be required to achieve the physical interconnection required. Fig.34 shows an example of a switch circuit that can be used to accomplish this interconnection for Fibre interconnects in which hardware signal detection (SDC) is used to activate the switch state changes.

[0139] In configurations in which the hosts implement multiple-redundant-pathing functionality, there is an alternate

host-side interconnect configuration that requires fewer interconnects to achieve similar levels of redundancy as shown in Fig.35. Note that host-side interconnects connecting an SVC to the hosts are not interconnected to the alternate SVC. In this configuration, interconnect redundancy is achieved by making each LMU accessible to the hosts over a host-side interconnect of one SVC also accessible through an alternate host-side interconnect on the alternate SVC. Should one of the interconnects break, become blocked, or otherwise malfunction, the hosts would still be able to access the LMU through the alternate SVC via the alternate interconnect. Similarly, should one of the SVCs malfunction, the other SVC can take over and, once again, the hosts would still be able to access the LMU through the normally-functioning SVC via the alternate interconnect.

[0140] A variation on the Redundant Serial ATA storage virtualization Subsystem uses Parallel ATA PSDs rather than Serial ATA PSDs. For each PSD, it incorporates a SATA-to-PATA conversion circuit that resides in close proximity to the PATA PSD between the access control switch and the PSD and typically together with the access control switch in the same FRU. This conversion circuit converts

SATA signals and protocol to PATA and back again in the opposite direction. The importance of a Redundant Serial ATA SVS that uses Parallel ATA PSDs lies in the fact that, in the short term, supplies of Serial ATA drives will still be relatively short compared to Parallel ATA and Serial ATA drives will still be significantly more expensive. During this transitional period, this kind of a subsystem would allow PATA PSDs to be substituted for SATA PSDs, eliminating the concerns over SATA PSD supply and cost. Such a subsystem would typically place the conversion circuit, together with the access control switch, in the removable canister in which the PSD resides. The removable canister allows the PSD and any related circuitry to be easily swapped out in the event of that a PSD and/or related circuitry needs servicing. By placing the conversion circuit in the canister, when SATA drives become readily available at a competitive price point, the entire canister contents can be swapped out with a SATA PSD and SATA related circuitry.

[0141] Please refer to Fig.36 and Fig.38. Fig.36 depicts a block diagram of a removable redundant SATA-PSD canister. Fig.37 includes more detailed block diagrams of the individual PCBs in the canister on which the access control

switch is situated. Fig.38 depicts a block diagram of a removable redundant PATA-PSD canister. Fig.39 includes more detailed block diagrams of the individual PCBs in the canister on which the access control switch and SATA-to-PATA conversion circuits are situated. They both have a pair SATA IO device interconnects and a set of access control switch control signals coming in from the two SVCs connecting into an access control switch circuit. The primary difference is in the presence of a SATA-to-PATA conversion circuit in the removable PATA-PSD canister which is otherwise absent in the removable SATA-PSD canister.

[0142] Those skilled in the art will readily observe that numerous modifications and alternations of the device may be made while retaining the teaching of the invention. Accordingly, the above disclosure should be construed as limited only by the metes and bounds of the appended claims.